Composition of Multi-dimensional Application Needs in Wireless Sensor Networks

Qi Han
Department of Mathematical and Computer Sciences
Colorado School of Mines, Golden, CO 80401
E-mail: qhan@mines.edu

1 Introduction

The emergence of micro-sensor nodes that integrate computation, networking, and sensing capabilities into a single device has created the possibility to build reactive systems that have the ability to monitor and react to physical events/phenomena. Given the importance and potential of the impact of sensor technologies, over the past decade, significant progress has been made on techniques to architect and program large-scale sensor systems. However, current research has primarily considered *functional* aspects of distributed sensor systems focusing on techniques to sense, capture, communicate, and compute over sensor networks. As sensor applications become more complex and diverse, *non-functional* application needs (such as timeliness, reliability, accuracy) become important and one application may even have several non-functional needs. As an illustrative example, consider a network of sensors monitoring ground movement to detect presence/arrival of enemy forces in a given region in a command and control application. Timeliness and reliability of sensing (in the presence of failures) might be of essence here if the countering maneuver requires immediate detection. Such timeliness and reliability requirements, however, come at certain costs, namely additional communication overheads, energy costs, etc. Furthermore, different applications over a given sensor infrastructure may have differing non-functional requirements. For instance, an online surveillance and actuation application might have real-time requirements, an analysis application over the same sensor system might only require that data be collected in a repository (eventually) at a given level of accuracy or spatial and temporal frequency. Such differing application requirements may pose competing requirements on the underlying sensor data collection, coordination, and storage mechanisms. For instance, from the perspective of the archival application, it might be both feasible and desirable that the data be collected, temporarily stored, compressed and then transmitted to the repository. A real-time monitoring/actuation application, however, may demand low latency. These non-functional needs are cross-cutting issues and better addressed by interactions among different layers: from medium access and networking to duty cycling, from data collection services to query services and applications layer.

2 Important Research Challenges

Often, sensor-based systems are built with narrow application goals in mind. We anticipate that as sensor network infrastructures become more sophisticated, they will have to accommodate several concurrent applications, some of which may have conflicting requirements in terms of timeliness, reliability and data accuracy. It is important for future sensor systems to accommodate alternative application types, and ensure that their conflicting requirements mesh with each other gracefully.

In order to cover a wide spectrum of sensor applications, we first differentiate between different application types based on the temporal aspect of sensor data: archival applications focus on historical (past) data, e.g., in order to detect patterns over time and build time-varying models. Real-time data collection is not critical here, but high quality and reliable archival of sensor-generated data is; monitoring applications are interested in current sensor values such as intrusion detection systems; and forecasting applications are interested in predicting future sensor values, where human operators involved in decision making processes can avail of information about trends in sensor values.

2.1 Specification and Translation of Application Needs

To ensure end-to-end support for applications with multiple non-functional needs, it is essential to provide a channel for applications to specify what they need and in what manner. Hence, we will need a high level declarative language, using which applications can specify both functional and non-functional needs. The specification of high level application needs will also be either translated to data needs or tradeoffs to guide sensing and collection planning.

Language Support: Sensor applications are often interested in data of certain type (e.g., temperature, or humidity data), and they are not interested in where and how the data is obtained. TinyDB uses a modified version of traditional SQL, focusing on issues related to when and how often data are acquired. We plan to further extend the query language of TinyDB and provide other clauses for users to specify non-functional needs: accuracy, energy efficiency, timeliness, and reliability.

Translation of Application Needs: In fact, an application's non-functional needs manifest themselves in the several layers of the system; by adapting and translating non-functional requirements between different layers in the system, we are able to satisfy the application requirements in a manner that is sensitive to the underlying resource availability. The complexity of mapping an application's non-functional needs to requirements over sensor measurements depends on both the application as well as the nature of the underlying sensor network. For instance, an accuracy requirement implies differently at application layer, query service layer and data collection layer. When we use acoustic sensors to track a moving object, the measurement at the sensor cannot directly translates to the displacement of the object. We should allow the application writers to adapt the application logic to deal with imprecise data without worrying about the underlying translation of data to measurement accuracy.

2.2 Architectures for Executing Sensor Applications

Given applications data needs, expressed in the form of a query, multiple execution architectures are possible. A traditional client-server approach would collect data at a (logically) centralized repository and evaluate queries over such a repository. Such an approach does not exploit the computation and storage capabilities available within the sensor network at sensor nodes. Recent work has seen a trend towards "in-network" computation that exploits resource at the sensors to trade computation for reduced communication. By computing directly in the sensor network, the data routing and computing can be co-optimized, resulting in higher scalability. There are, however, limitations of such an approach including limitations on types of data access that can be supported in-network, complexity of optimally splitting computation between sensors and servers, no direct way of exploiting application's tolerance to errors and faults. Yet another approach is a hierarchical view in which the actual placement of the distributed server functionality is a function of the node capabilities in a hierarchical setting, the architecture allowing for dynamic migration of functionality based on the well-developed client server model. The exploration of such architectures has, in the literature, been motivated from a narrow perspective of suitability for one (or more) sensor application scenarios (e.g., monitoring, archiving), a comprehensive understanding of the suitability and feasibility of diverse architectures under different situations and blend of application loads is missing. Furthermore, what is entirely missing is the

implications of supporting non-functional requirements over these architectures. Therefore, there is a need to develop a thorough comprehensive analysis of possible architectures with the focus on supporting non-functional application requirements. Appropriate data model and data placement strategies also need to be explored in the architecture.

2.3 Development of Composition Techniques

The goal here is to address the specific issue of cross-layer composability, i.e., processing of queries while ensuring the multiple non-functional needs posed by application queries are met in a cost-effective manner. However, application requirements may conflict with each other. Consider the following cases. (a) accuracy vs. timeliness: Frequent updates from sensors will undoubtedly improve accuracy, however, it might also leave the server not enough time to process user queries, hence violating many time constraints. (b) timeliness vs. reliability: In order to ensure reliability in the presence of faults, there may be a need to re-transmit data; however, this recovery can be time-consuming, hence leading to degradation in timeliness satisfaction. (c) reliability vs. accuracy: queries may derive some missing sensor data based on historical or neighboring reports, and this might be sufficient for reliability needs; however, this implies that the derived data may not be accurate. Therefore, arbitrary composition of those techniques developed for addressing each individual non-functional need will result in undesirable system performance. To ensure judicious composition, several general strategies can be exploited: error-aware prediction, spatiotemporal sensor data correlation, and application-aware caching. Finally, we should support two complementary types of sensor applications.

- Maximize certain combination of performance metrics (timeliness, reliability, or accuracy) without exceeding energy budget: with the finite remaining energy level on each sensor node, we are faced with a joint optimization problem when the objective of an application is to maximize more than one metric (i.e., two or three among reliability, timeliness, accuracy).
- Minimize energy consumption while achieving a minimum acceptable level of performance metrics: the multiple constraints can be any combination of requirements for timeliness, reliability and accuracy.